DECENTRALIZED EXCHANGE
Smart Contract
Security Review

November 7, 2020

Prepared by	Reviewed by	Prepared for
Pongsabutra Viraseranee Guillaume Le Saint ATATO	Khun Nattapon Nimakul CTO, Kulap.io	Audit Committee Kulap.io

Background

Version History

Version number	Date	Author	Comments
Revision 1	03/10/2020	atato	Preliminary version
Revision 2	04/10/2020	atato	Added names of authors, fixed typos
Revision 3	07/11/2020	atato	Updated report after Kulap took corrective actions.

Objectives

Kulap.io is developing a decentralized digital assets exchange and has applied for a digital asset Broker license under the definitions of The Kingdom of Thailand's Royal Decree on Digital Assets Business. As part of the broker license attribution by the Securities and Exchange Commission of Thailand, a security assessment of some of the Ethereum smart contracts should be conducted. Kulap.io has contracted atato to conduct the initial security review.

Review Report

Atato has produced a security review of kulap.io's decentralized exchange smart contracts, using vulnerability assessment techniques, and a review of smart contract security best practices.
The reference audited contract is available on GitHub at the following address:

- https://github.com/kulapio/dex-smart-contract/blob/deab1f6c0d3b66056fb562a57bc031f38356b67d/contracts/KULAPDex.sol

Report Overview

Atato has reviewed smart contracts against ConsenSys Diligence and Open Zeppelin security best practices as well as known security issues using the Smart Contract Weakness Classification and Test Cases. Out of these, there were 0 high-priority security findings, 1 medium-priority finding, and 11 low-priority findings. Atato has provided a recommended action plan for each.

Table of Contents

Review Methodology

Overview

The security review conducted doesn't replace a full security audit of the overall Kulap.io technology infrastructure. Its scope is limited to the KULAPDex.sol smart contract, and to some aspects of the smart contract itself. Security best practices strongly recommend that Kulap.io conduct a full security audit of the on-chain and off-chain components of their infrastructure, and the interaction between the two.

Scope

The security review covers the following components of the Kulap.io platform:

- Kulap.io smart contracts, in particular:
  - KULAPDex.sol
  - Commit deab1f6
  - Reference file
  - https://github.com/kulapio/dex-smart-contract/blob/deab1f6c0d3b66056fb562a57bc031f38356b67d/contracts/KULAPDex.sol
- Imported associated smart contracts, in particular OpenZeppelin smart contracts:
  - openzeppelin-solidity/contracts/utils/ReentrancyGuard.sol
  - openzeppelin-solidity/contracts/math/SafeMath.sol
  - openzeppelin-solidity/contracts/ownership/Ownable.sol
  - Commit 58a3368
  - Reference tree https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v2.5.0
- Kulap.io helpers and interfaces, in particular:
  - ./helper/ERC20Interface.sol
  - ./interfaces/IKULAPTradingProxy.sol
  - ./interfaces/IKULAPDex.sol
  - Commit deab1f6
  - Reference tree
  - https://github.com/kulapio/dex-smart-contract/tree/deab1f6c0d3b66056fb562a57bc031f38356b67d/contracts
- Compilation and testing environment, in particular:
  - .mocharc.json
  - .waffle.json

  - package.json
  - Commit deab1f6
  - Reference tree
  - https://github.com/kulapio/dex-smart-contract/tree/deab1f6c0d3b66056fb562a57bc031f38356b67d/contracts

The security review covers the following:

- Solidity best practices, including:
  - Documentation
  - Linting
  - Compiler warnings
  - Unused code sections
  - Todo comments
  - Test instructions
  - Tests execution
  - Testing dependencies
- Automated analysis, including:
  - Assertions and property checking
  - Byte-code safety
  - Authorization controls
  - Control flow
  - ERC standards compliance
  - Solidity coding best practices

Tools

The following tools and material were used in conducting the review:

- Smart Contract Weakness Classification and Test Cases
- ConsenSys Smart Contract Best Practices
- MythX Professional Edition Subscription
- MythX Python CLI

Recommendations

#	Priority	Finding	Action Plan
1	Medium	Multiple calls are executed in the same transaction	A1
2	Low	A floating pragma is set	A2
3	Low	Read of persistent state following external call (1 out of 2)	A3
4	Low	Read of persistent state following external call (2 out of 2)	A3
5	Low	A call to a user-supplied address is executed	A4
6	Low	Dependencies do not include OpenZeppelin	A5

7	Low	No OpenZeppelin version is specified	A5
8	Low	Imports of OpenZeppelin does not use scoped NPM packages	A6
9	Low	Compile command incorrect	A8
10	Low	Compile fails	A5
11	Low	Test fails	A5
12	Low	Cannot measure the test coverage	A7

Note that to conduct the security analysis, OpenZeppelin 2.5.0 was specified and used. See findings F6, F7, F9, F10, F11, and action plan A6 for more details.

These recommendations and proposed action plans are detailed further below.

Review Findings

Overview

High

There were no high severity findings in the review.

Medium

F1 SWC-113 Multiple calls are executed in the same transaction

- File: KULAPDex.sol
- Location L:316 C:30
- https://github.com/kulapio/dex-smart-contract/blob/deab1f6c0d3b66056fb562a57bc031f38356b67d/contracts/KULAPDex.sol#L316
-
```
314             srcAmountBefore = address(this).balance;
315         } else {
316             srcAmountBefore = _src.balanceOf(address(this));
```
- This call is executed following another call within the same transaction. The call may never get executed if a prior call fails permanently. This might be caused intentionally by a malicious callee. If possible, refactor the code such that each transaction only executes one external call or make sure that all callees can be trusted (i.e. they're part of your own codebase).
- https://swcregistry.io/docs/SWC-113

Low

F2 SWC-103 A floating pragma is set

- File: KULAPDex.sol
- Location L:1 C:0
- https://github.com/kulapio/dex-smart-contract/blob/deab1f6c0d3b66056fb562a57bc031f38356b67d/contracts/KULAPDex.sol#L1
-
```
1  pragma solidity ^0.5.12;
```
- The current pragma Solidity directive is "^0.5.12". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.
- https://swcregistry.io/docs/SWC-103

F3 SWC-107 Read of persistent state following external call (1 out of 2)

- File: KULAPDex.sol
- Location L:47 C:16
- https://github.com/kulapio/dex-smart-contract/blob/deab1f6c0d3b66056fb562a57bc031f38356b67d/contracts/KULAPDex.sol#L47
-
```
45
46      modifier onlyTradingProxyEnabled(uint _index) {
47          require(tradingProxies[_index].enable == true, "This trading proxy is disabled");
```
- The contract account state is accessed after an external call. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.
- https://swcregistry.io/docs/SWC-107

F4 SWC-107 Read of persistent state following external call (2 out of 2)

- File: KULAPDex.sol
- Location L:313 C:12
- https://github.com/kulapio/dex-smart-contract/blob/deab1f6c0d3b66056fb562a57bc031f38356b67d/contracts/KULAPDex.sol#L313
-
```
312
313         if (etherERC20 == _src) { // Source
```
- The contract account state is accessed after an external call. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.
- https://swcregistry.io/docs/SWC-107

F5 SWC-107 A call to a user-supplier address is executed

- File: KULAPDex.sol
- Location L:607 C:8
- https://github.com/kulapio/dex-smart-contract/blob/deab1f6c0d3b66056fb562a57bc031f38356b67d/contracts/KULAPDex.sol#L607
-
```
606   {
607       token.transfer(msg.sender, amount);
```
- An external message call to an address specified by the caller is executed. Note that the callee account might contain arbitrary code and could re-enter any function within this contract. Reentering the contract in an intermediate state may lead to unexpected behavior. Make sure that no state modifications are executed after this call and/or reentrancy guards are in place.
- https://swcregistry.io/docs/SWC-107

F6 Dependencies do not include OpenZeppelin

- File: package.json
- Location: L9 C:3
- https://github.com/kulapio/dex-smart-contract/blob/deab1f6c0d3b66056fb562a57bc031f38356b67d/package.json#L9
-
```
9   "devDependencies": {
10    "@types/chai": "^4.2.6",
11    "@types/mocha": "^5.2.7",
12    "chai": "^4.2.0",
13    "ethereum-waffle": "^2.4.1",
14    "ethereumjs-util": "^6.2.0",
15    "mocha": "^6.2.2",
16    "prettier": "^1.19.1",
17    "rimraf": "^3.0.0",
18    "solc": "0.5.16",
19    "ts-node": "^8.5.4",
20    "typescript": "^3.7.3"
21  },
```
- The package definition does not include the OpenZeppelin library. This also causes issues found in F7, F10, and F11.

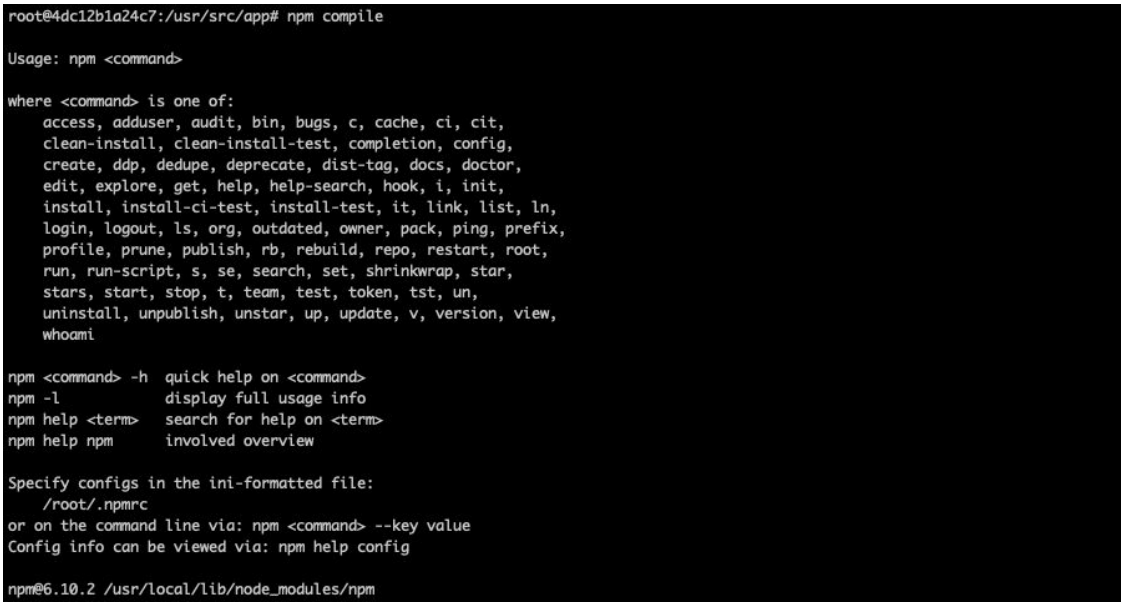
F7 No OpenZeppelin version is specified

- File: package.json
- Location: L9 C:3
- https://github.com/kulapio/dex-smart-contract/blob/deab1f6c0d3b66056fb562a57bc031f38356b67d/package.json#L9
-
```
9   "devDependencies": {
10    "@types/chai": "^4.2.6",
11    "@types/mocha": "^5.2.7",
12    "chai": "^4.2.0",
13    "ethereum-waffle": "^2.4.1",
14    "ethereumjs-util": "^6.2.0",
15    "mocha": "^6.2.2",
16    "prettier": "^1.19.1",
17    "rimraf": "^3.0.0",
18    "solc": "0.5.16",
19    "ts-node": "^8.5.4",
20    "typescript": "^3.7.3"
21  },
```
- The package definition does not set a version for the OpenZeppelin library. This also causes issues found in F7, F10, and F11.

F8 Imports of OpenZeppelin does not use scoped NPM packages

- File: KULAPDex.sol
- Location: L3 C9, L4 C9, L5 C9
- https://github.com/kulapio/dex-smart-contract/blob/deab1f6c0d3b66056fb562a57bc031f38356b67d/contracts/KULAPDex.sol#L3
- 
```
3  import 'openzeppelin-solidity/contracts/utils/ReentrancyGuard.sol';
4  import "openzeppelin-solidity/contracts/math/SafeMath.sol";
5  import "openzeppelin-solidity/contracts/ownership/Ownable.sol";
6  import "./helper/ERC20Interface.sol";
```
- The import of open-zeppelin contracts does not use scoped packages. Scoped packages are published by the issuing organization.

F9 Compile command incorrect

- File: readme.md
- Location L:26 C:7
- https://github.com/kulapio/dex-smart-contract/tree/deab1f6c0d3b66056fb562a57bc031f38356b67d#compile
- 
```
root@4dc12b1a24c7:/usr/src/app# npm compile

Usage: npm <command>

where <command> is one of:
    access, adduser, audit, bin, bugs, c, cache, ci, cit,
    clean-install, clean-install-test, completion, config,
    create, ddp, dedupe, deprecate, dist-tag, docs, doctor,
    edit, explore, get, help, help-search, hook, i, init,
    install, install-ci-test, install-test, it, link, list, ln,
    login, logout, ls, org, outdated, owner, pack, ping, prefix,
    profile, prune, publish, rb, rebuild, repo, restart, root,
    run, run-script, s, se, search, set, shrinkwrap, star,
    stars, start, stop, t, team, test, token, tst, un,
    uninstall, unpublish, unstar, up, update, v, version, view,
    whoami

npm <command> -h  quick help on <command>
npm -l            display full usage info
npm help <term>   search for help on <term>
npm help npm      involved overview

Specify configs in the ini-formatted file:
    /root/.npmrc
or on the command line via: npm <command> --key value
Config info can be viewed via: npm help config

npm@6.10.2 /usr/local/lib/node_modules/npm
```
- The compile command ‘npm compile’ fails. The compile script must be started by ‘npm run compile’. The documentation should be updated.
- The error can be reproduced in a docker container running node:12, the version recommended in the readme.

F10 Compile fails

- File: readme.md
- Location L:26 C:7
- https://github.com/kulapio/dex-smart-contract/tree/deab1f6c0d3b66056fb562a57bc031f38356b67d#compile

```
root@4dc12b1a24c7:/usr/src/app# npm run compile

> dex-smart-contract@1.0.0 precompile /usr/src/app
> yarn clean

yarn run v1.17.3
$ rimraf ./build/
Done in 0.22s.

> dex-smart-contract@1.0.0 compile /usr/src/app
> waffle .waffle.json

Error: None of the sub-resolvers resolved "openzeppelin-solidity/contracts/utils/ReentrancyGuard.sol" location.
    at resolverError (/usr/src/app/node_modules/@resolver-engine/core/build/resolverengine.js:75:32)
    at ResolverEngine.<anonymous> (/usr/src/app/node_modules/@resolver-engine/core/build/resolverengine.js:52:23)
    at Generator.next (<anonymous>)
    at fulfilled (/usr/src/app/node_modules/@resolver-engine/core/build/resolverengine.js:4:58)
    at processTicksAndRejections (internal/process/task_queues.js:85:5)
npm ERR! code ELIFECYCLE
npm ERR! errno 1
npm ERR! dex-smart-contract@1.0.0 compile: `waffle .waffle.json`
npm ERR! Exit status 1
npm ERR!
npm ERR! Failed at the dex-smart-contract@1.0.0 compile script.
npm ERR! This is probably not a problem with npm. There is likely additional logging output above.

npm ERR! A complete log of this run can be found in:
npm ERR!     /root/.npm/_logs/2020-10-02T11_53_56_616Z-debug.log
root@4dc12b1a24c7:/usr/src/app# npm run test

> dex-smart-contract@1.0.0 pretest /usr/src/app
> yarn compile

yarn run v1.17.3
$ yarn clean
$ rimraf ./build/
$ waffle .waffle.json
Error: None of the sub-resolvers resolved "openzeppelin-solidity/contracts/utils/ReentrancyGuard.sol" location.
    at resolverError (/usr/src/app/node_modules/@resolver-engine/core/build/resolverengine.js:75:32)
    at ResolverEngine.<anonymous> (/usr/src/app/node_modules/@resolver-engine/core/build/resolverengine.js:52:23)
    at Generator.next (<anonymous>)
    at fulfilled (/usr/src/app/node_modules/@resolver-engine/core/build/resolverengine.js:4:58)
    at processTicksAndRejections (internal/process/task_queues.js:85:5)
error Command failed with exit code 1.
info Visit https://yarnpkg.com/en/docs/cli/run for documentation about this command.
npm ERR! code ELIFECYCLE
npm ERR! errno 1
npm ERR! dex-smart-contract@1.0.0 pretest: `yarn compile`
npm ERR! Exit status 1
npm ERR!
npm ERR! Failed at the dex-smart-contract@1.0.0 pretest script.
npm ERR! This is probably not a problem with npm. There is likely additional logging output above.

npm ERR! A complete log of this run can be found in:
npm ERR!     /root/.npm/_logs/2020-10-02T11_55_46_350Z-debug.log
```

- The compile command ‘npm run compile’ fails.
- The error can be reproduced in a docker container running node:12, the version recommended in the readme.

F11 Test fails

- File: readme.md
- Location L:29 C:7
- https://github.com/kulapio/dex-smart-contract/tree/deab1f6c0d3b66056fb562a57bc031f38356b67d#test

```
root@4dc12b1a24c7:/usr/src/app# npm test

> dex-smart-contract@1.0.0 pretest /usr/src/app
> yarn compile

yarn run v1.17.3
$ yarn clean
$ rimraf ./build/
$ waffle .waffle.json
Error: None of the sub-resolvers resolved "openzeppelin-solidity/contracts/utils/ReentrancyGuard.sol" location.
    at resolverError (/usr/src/app/node_modules/@resolver-engine/core/build/resolverengine.js:75:32)
    at ResolverEngine.<anonymous> (/usr/src/app/node_modules/@resolver-engine/core/build/resolverengine.js:52:23)
    at Generator.next (<anonymous>)
    at fulfilled (/usr/src/app/node_modules/@resolver-engine/core/build/resolverengine.js:4:58)
    at processTicksAndRejections (internal/process/task_queues.js:85:5)
error Command failed with exit code 1.
info Visit https://yarnpkg.com/en/docs/cli/run for documentation about this command.
npm ERR! Test failed.  See above for more details.
```

- The test command 'npm test' fails.
- The error can be reproduced in a docker container running node:12, the version recommended in the readme.
- Note that after applying the recommendation A5 Add missing OpenZeppelin dependency, tests run successfully.

F12 Cannot measure the tests coverage

- File: .waffle.json
- Location: N/A
- It is recommended to measure the coverage of smart contracts tests, to ensure that each statement of the contract is hit at least once by tests.
- To our knowledge, there are no Waffle test coverage measurement tools available today.

Action Plan

Overview

A total of 8 action plan items are suggested to address the issues found and to follow best practices with regards to operations of the KULAPDEx.sol smart contract.
As stated in the report methodology overview, atato recommends that a full security audit of the Kulap.io platform be planned to evaluate the components that were not in the scope of this review.

Details

A8 Other recommendations

Recommendation: Update the documentation

- Associated finding: F9
- A minor update of the package's compile command is needed to fix a typo.

Recommendation: Update the documentation

- Associated finding: N/A
- To prepare for a full audit, it is recommended to provide documentation that follows best practices.
- Reference best practice:
- https://consensys.github.io/smart-contract-best-practices/documentation_procedures/

Recommendation: Make contract pausable
- Associated finding: N/A
- To have the ability to suspend execution of the contract, where an unplanned event would require Kulap.io to do so, making the contract pausable should be considered.
- Reference best practice:
- https://docs.openzeppelin.com/contracts/3.x/api/utils#Pausable

Recommendation: Make contract upgradeable
- Associated finding: N/A
- To have the ability to modify the contract, where a bug, fix or feature would require Kulap.io to do so, making the contract upgradeable should be considered.
- Reference best practice:
- https://docs.openzeppelin.com/upgrades/2.8/creating-upgradeable-from-solidity

A1 Favor pull over push for external calls

- Associated findings: F1
- External calls can fail accidentally or deliberately. To minimize the damage caused by such failures, it is often better to isolate each external call into its own transaction that can be initiated by the recipient of the call. This is especially relevant for payments, where it is better to let users withdraw funds rather than push funds to them automatically. (This also reduces the chance of problems with the gas limit.) Avoid combining multiple ether transfers in a single transaction.
- Reference best practice:
- https://consensys.github.io/smart-contract-best-practices/recommendations/#favor-pull-over-push-for-external-calls

A2 Set pragma version

- Associated findings: F2
- Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.
- Reference best practice:
- https://consensys.github.io/smart-contract-best-practices/recommendations/#lock-pragmas-to-specific-compiler-version

A3 Avoid state changes after external calls

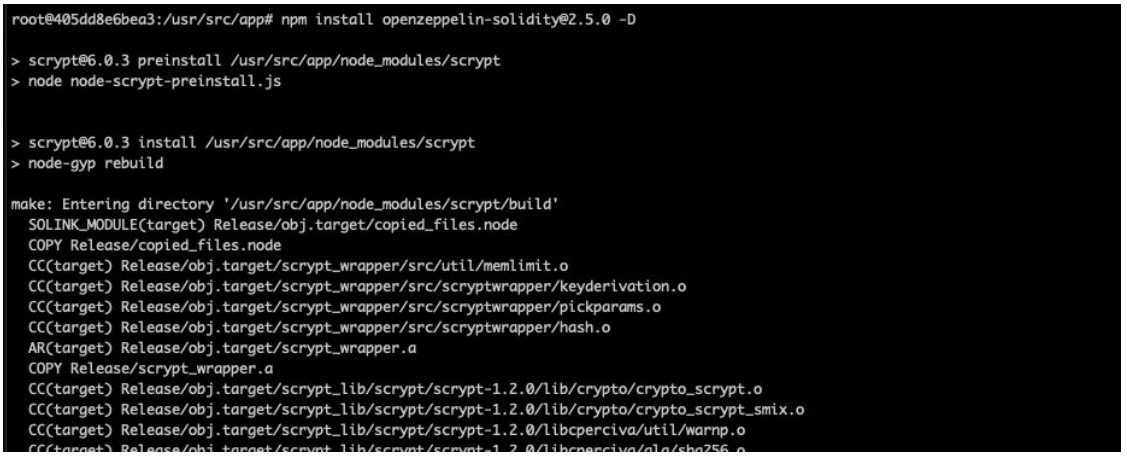
- Associated findings: F3, F4
- Whether using raw calls (of the form someAddress.call()) or contract calls (of the form ExternalContract.someMethod()), assume that malicious code might execute. Even if ExternalContract is not malicious, malicious code can be executed by any contracts it calls. One particular danger is malicious code may hijack the control flow, leading to vulnerabilities due to reentrancy. (See Reentrancy for a fuller discussion of this problem). If you are making a call to an untrusted external contract, avoid state changes after the call. This pattern is also sometimes known as the checks-effects-interactions pattern.
- Reference best practice:

- https://consensys.github.io/smart-contract-best-practices/recommendations/#avoid-state-changes-after-external-calls

A4 Use caution when making external calls

- Associated findings: F3, F4
- Calls to untrusted contracts can introduce several unexpected risks or errors. External calls may execute malicious code in that contract or any other contract that it depends upon. As such, every external call should be treated as a potential security risk. When it is not possible, or undesirable to remove external calls, use the recommendations in the rest of this section to minimize the danger.
- Reference best practice:
- https://consensys.github.io/smart-contract-best-practices/recommendations/#use-caution-when-making-external-calls

A5 Add missing OpenZeppelin dependency

- Associated findings: F6, F7, F10, F11
- OpenZeppelin is not part of the package definition. Therefore, a version of the OpenZeppelin library is not specified either. From the names and locations of the contract used, we think that the intention was to use version 2 of the OpenZeppeling library. (for example /contracts/ownership/Ownable.sol is imported, which is the correct path for version 2, but the contract has been moved to /contracts/access/Ownable.sol in version 3).
- To conduct the audit, we used the latest stable release branch of the version, v2.5.0.
- 

```
root@405dd8e6bea3:/usr/src/app# npm install openzeppelin-solidity@2.5.0 -D

> scrypt@6.0.3 preinstall /usr/src/app/node_modules/scrypt
> node node-scrypt-preinstall.js


> scrypt@6.0.3 install /usr/src/app/node_modules/scrypt
> node-gyp rebuild

make: Entering directory '/usr/src/app/node_modules/scrypt/build'
  SOLINK_MODULE(target) Release/obj.target/copied_files.node
  COPY Release/copied_files.node
  CC(target) Release/obj.target/scrypt_wrapper/src/util/memlimit.o
  CC(target) Release/obj.target/scrypt_wrapper/src/scryptwrapper/keyderivation.o
  CC(target) Release/obj.target/scrypt_wrapper/src/scryptwrapper/pickparams.o
  CC(target) Release/obj.target/scrypt_wrapper/src/scryptwrapper/hash.o
  AR(target) Release/obj.target/scrypt_wrapper.a
  COPY Release/scrypt_wrapper.a
  CC(target) Release/obj.target/scrypt_lib/scrypt/scrypt-1.2.0/lib/crypto/crypto_scrypt.o
  CC(target) Release/obj.target/scrypt_lib/scrypt/scrypt-1.2.0/lib/crypto/crypto_scrypt_smix.o
  CC(target) Release/obj.target/scrypt_lib/scrypt/scrypt-1.2.0/libcperciva/util/warnp.o
```

package.json

```
@@ -9,6 +9,7 @@
   "devDependencies": {
     "@types/chai": "^4.2.6",
     "@types/mocha": "^5.2.7",
+    "openzeppelin-solidity": "2.5.0",
     "chai": "^4.2.0",
     "ethereum-waffle": "^2.4.1",
     "ethereumjs-util": "^6.2.0",
```

- https://github.com/OpenZeppelin/openzeppelin-contracts/tree/release-v2.5.0.
- As open OpenZeppelin maintains a stable contract API, we recommend an update to the latest stable release version, v3.2.0.
- https://github.com/OpenZeppelin/openzeppelin-contracts/releases/tag/v3.2.0

- Reference best practice:
- https://docs.openzeppelin.com/contracts/3.x/releases-stability#api-stability

A6 Import using scoped packages

- Associated findings: F8
- OpenZeppelin contracts are imported using non-scoped packages names. The recommended way to import contracts is to use scoped packages (@organization).
- 

- 

- Reference best practice:
- https://docs.npmjs.com/using-npm/scope.html
- https://docs.openzeppelin.com/contracts/3.x/#usage

A7 Migrate tests to Buidler to measure coverage

- Associated findings: F12
- The testing framework used doesn't allow, to our knowledge, for integration with code coverage measurement tools. Migrating to Buidler would allow measuring the test coverage.
- Reference best practice:
- https://blog.colony.io/code-coverage-for-solidity-eecfa88668c2/
- https://buidler.dev/guides/waffle-testing.html#migrating-an-existing-waffle-project

Remediation

Overview

Kulap.io has taken corrective measures to address the findings of this report.
The latest commit analyzed is commit 585d3ba

- https://github.com/kulapio/dex-smart-contract/commit/585d3bafab6b1579c31eb0eb8bdca27a94459f69

Kulap.io has analyzed the report recommendations, and applied the action plan for most of the findings. For some of the findings, the associated risks were considered acceptable by Kulap.io when evaluated in relation to the overall design of their solution, in particular as it is non-custodial.

Findings

F1 - Remains

The referenced smart contract calls are made on a function of the 'view' type (balanceOf).
It should also be noted that the KULAPDex contract operates in a non-custodial way.

F2 - Solved

A fixed pragma is now used.

- https://github.com/kulapio/dex-smart-contract/blob/585d3bafab6b1579c31eb0eb8bdca27a94459f69/contracts/KULAPDex.sol#L1

F3, F4 - Remains

The referenced account state (tradingProxies, etherERC20) are read but not modified.

F5 - Remains

The associated function (collectRemainingToken) has the onlyOwner modifier applied.

F6, F7, F8, F10, F11 - Solved

OpenZeppelin 2.5.0 is added to the dependencies.

- https://github.com/kulapio/dex-smart-contract/blob/585d3bafab6b1579c31eb0eb8bdca27a94459f69/package.json#L16

Scoped packages are used for the imports.

- https://github.com/kulapio/dex-smart-contract/blob/585d3bafab6b1579c31eb0eb8bdca27a94459f69/contracts/KULAPDex.sol#L3

F9 - Solved

The compile command documentation has been updated.

- https://github.com/kulapio/dex-smart-contract/blob/585d3bafab6b1579c31eb0eb8bdca27a94459f69/README.md

F10 - Remains

The test coverage still cannot be measured.